FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Akseli Tyvelä

VERSION CONTROL AND AUTOMATED FILTERING OF SIMULATION DATA TO SUPPORT DESIGN FOR MANUFACTURABILITY IN VIRTUAL NEW PRODUCT INTRODUCTION

Master's Thesis
Degree Programme in Computer Science and Engineering
June 2021

Tyvelä A. (2021) Version Control and Automated Filtering of Simulation Data to Support Design for Manufacturability in Virtual New Product Introduction. University of Oulu, Degree Programme in Computer Science and Engineering, 42 p.

ABSTRACT

Design for Manufacturability methodology has been successfully applied to new product introduction process for a number of years to bring down the manufacturing costs and to shorten the time-to-market. Still, physical prototypes have been needed to ensure that parts are within set tolerances and can be fitted together as designed. A successful new product introduction process can require multiple prototyping rounds, and when crafting a single physical prototype is costly, the whole process will become very expensive and time consuming.

To further counter the cost of physical prototypes and to reduce time to market, physical prototyping rounds can be reduced by implementing virtual prototyping rounds by analyzing 3D mechanics with simulation tools. For the simulations to be valid, the results must be able to be verified and the data managed so that the information gathered by analyzing the 3D mechanics can be used in re-design in next prototyping rounds.

In this thesis a software system was developed to support the virtual new product introduction process. The system includes a machine learning model for identifying falsely detected collisions in resulting simulation data. A database system and graphical user interface were implemented for storing and version controlling the simulation results. The implemented machine learning based filtering system was tested and its performance measured against real-life examples. The evaluation results show great promise for the implemented model to be used as a tool to reduce the amount of falsely detected collisions. However, for some of the tested examples, performance remained low, indicating the need for further development, especially in terms of the amount and quality of data used for training the machine learning model.

Keywords: Design for Manufacturability, machine learning, new product introduction, version control

Tyvelä A. (2021) Simulaatiotietojen versionhallinta ja automaattinen suodatus tukemaan valmistettavuuden suunnittelua virtuaalisessa uuden tuotteen käyttöönotossa. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 42 s.

TIIVISTELMÄ

Vaikka valmistettavuuden suunnittelua on jo vuosien ajan käytetty uuden tuotteen käyttöönottoprosessin aikana alentamaan tuotteen valmistuskustannuksia ja nopeuttamaan sen saamista markkinoille, uuden tuotteen käyttöönotossa tarvitaan edelleen fyysisten prototyyppien valmistusta, jotta voidaan taata tuotteen eri komponenttien yhteensopivuus. Jotta uuden tuotteen käyttöönotto voidaan tehdä onnistuneesti, vaatii se usein useiden fyysisten prototyyppien valmistusta, mikä tekee prosessista kalliin ja aikaa vievän.

Fyysisten prototyyppien tilalla voidaan implementoida virtuaalisia prototyyppikierroksia joko osittain tai kokonaan korvaten fyysiset prototyypit. Virtuaaliset prototyyppikierrokset voidaan tehdä esimerkiksi analysoimalla tuotteiden 3D-malleja simulointiohjelmistoilla. Jotta simulointeja voidaan käyttää menestyksekkäästi, täytyy simulaatiotulosten olla todennettavissa ja tulosdatan olla hallittavissa niin että sitä voidaan hyödyntää uudelleen suunnittelussa seuraavan prototyyppikierroksen aikana.

Tässä työssä kehitettiin ohjelmisto tukemaan valmistettavuuden suunnittelua virtuaalisessa uuden tuotteen käyttöönotossa. Ohjelmisto sisältää koneoppimispohjaisen järjestelmän simulaatiotulosten varmistamiseksi, jolla voidaan tunnistaa väärin havaittuja törmäyksiä simulaatiodatan joukosta. Tämän lisäksi ohjelmisto sisältää käyttöliittymän ja tietokannan, jolla simulaatiotuloksia voidaan säilöä versionhallinnan tapaan.

Työssä kehitettyä koneoppimispohjaista suodatusjärjestelmää testattiin ja sen suorituskykyä arvioitiin suodattamalla virheitä sisältäviä simulaatiotuloksia. Arviointitulokset osoittavat, että kehitettyä järjestelmää voitaisiin käyttää työkaluna simulaatiotulosten suodattamiseen. Joidenkin testattujen esimerkkien kohdalla suorituskyky kuitenkin jäi alhaiseksi, mikä osoittaa tarpeen myös jatkokehitykselle erityisesti koneoppimismallin koulutukseen käytetyn datan määrän ja laadun osalta.

Avainsanat: valmistettavuuden suunnittelu, koneoppiminen, uuden tuotteen käyttöönotto, versionhallinta

TABLE OF CONTENTS

FOREWORD

First, I would like to thank Janne Piilola, my line manager at Nokia Mobile Networks, for initially hiring me and by doing so making this thesis possible. I would also like to thank everyone else at Nokia Mobile Networks that has contributed to this project, especially my supervisor Ari Teppo along with the whole project steering group.

I would also want to thank my supervisors, D.Sc. Lauri Tuovinen and D.Sc. Jaakko Suutala for supervising and reviewing my thesis. Special thanks for the work they put in towards the end when time was of the essence.

Moreover, I would like to thank my friend Milja for crucial tips during this project and for listening to me complain when things didn't go as planned.

Finally, I would like to thank my family, friends, and my girlfriend Siiri for supporting me during my studies and especially this project.

Oulu, June 10th, 2021

Akseli Tyvelä

LIST OF ABBREVIATIONS AND SYMBOLS

3D	Three Dimensional
AI	Artificial Intelligence
API	Application Programming Interface
CAD	Computer Aided Design
CVCS	Centralized Version Control System
DB	Database
DFA	Design for Assembly
DFC	Design for Cost
DFM	Design for Manufacturability
DFMA	Design for Manufacturability and Assembly
DFX	Design for X or Design for Excellence
DVCS	Distributed Version Control System
FReMP	Flask, ReactJs, MongoDB and Python
JSON	JavaScript Object Notation
ML	Machine Learning
NoSQL	Non-SQL or Non-Relational
NPD	New Product Development
NPI	New Product Introduction
REST	Representational State Transfer
sklearn	Scikit-Learn
SVM	Support Vector Machine
nu	An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors

1. INTRODUCTION

Design for X (DFX) is an engineering practice which manages guidelines on how different properties of products can be reached by considering them early in the product development process.. The variable X refers to different applications of DFX. These applications can be different virtues (such as cost) that the product is set to fulfill or they can focus on a certain life phase of the product's life cycle (such as assembly).[1]

One of the most used and studied DFX application is Design for Manufacturability (DFM) [2]. DFM aims to design a product in such a way that it would be as easy to manufacture as possible to drive down the manufacturing cost and to shorten time-to-market [3]. Traditionally DFM has provided its support in the form of both general and manufacturing process specific guidelines [2]. While the surrounding world and different software solutions have evolved, the DFM analysis has remained as a manual process [4]. More recent development has given birth to feature-based software tools that can be used for simulating and analysing three dimensional (3D) computer aided design (CAD) models to shorten the time required for DFM analysis and for reducing workload and need for resources [4].

Machine learning (ML) is a subfield of artificial intelligence (AI) that refers to computing algorithms that are used to learn from input data to perform different kinds of tasks such as classification [5]. Machine learning can be divided into three subcategories: supervised, unsupervised and reinforcement learning [6]. In this thesis work a unsupervised novelty detection algorithm, one-class support vector machine, is implemented to filter the simulation results.

1.1. New Product Introduction

As competition has been increasing in the global market, companies need to answer this competition by developing better products [7, 8]. New product introduction (NPI), also know ans new product development (NPD), has for a long time been one of the core functions of the companies, especially in companies that manufacture products with short life cycles [7].

NPI is a process which translates a product idea or a concept into a complete, physical product. NPI process includes four primary stages that are shown in Figure 1. After each of the stages, the current stage of the process is evaluated, and based on the evaluation the project can either move forward to the next stage, or it can be recycled back into the previous stage or completely discarded [9].

This thesis work is done as a part of a larger virtualization project which has the goal of extending the stage 2 of simulating the product design and manufacturing process until the production is reached in phase 4. By covering the stage 3 fully or partly with simulating, manufacturing of physical prototypes could be avoided, or at least the amount of prototypes reduced, which would reduce cost and time-to-market of the product.

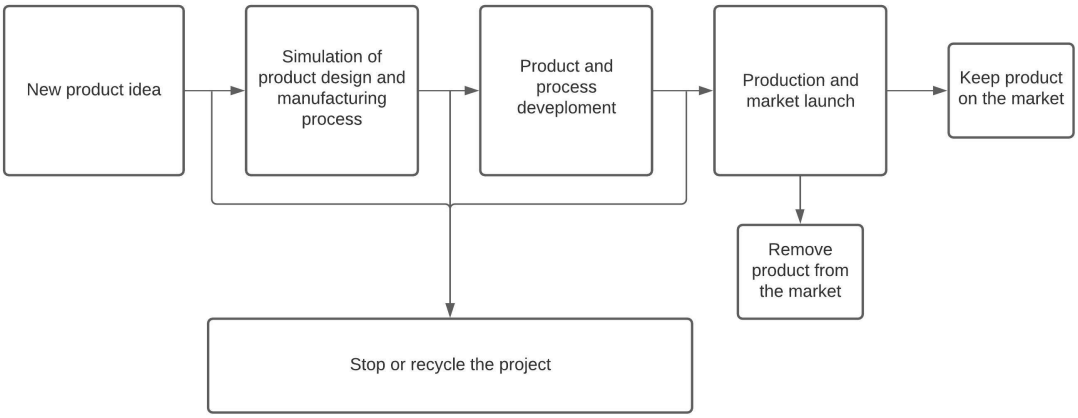

Figure 1. New product introduction process. Adapted from [9].

1.2. Motivation

When moving from NPI to virtual NPI process, physical prototyping rounds are replaced fully, or at least partly, with virtual prototyping rounds to reduce or eliminate the cost of manufacturing physical prototypes. Removing or reducing physical prototypes relies on a functioning simulation process that can supersede physical prototypes and offer the same insights that physical prototypes would.

For the simulation process to be functional, the simulation software and process need to be proven correct, and simulations results themselves need to be validatable so that the data gathered through simulations is usable and can be fed back to designers for the next prototyping round.

This thesis was made as a part of a major company's larger software test project. During the project a new feature-based analysis software tool was tested in order to find out if it could be actually used as a simulation tool for analysing 3D CAD models to support Design for Manufacturability in a larger virtualization project, which aims at moving the whole new product introduction process into virtual form.

While the software tested provided useful tools for highlighting design errors in 3D models, some drawbacks were identified as well, especially regarding the collision detection feature used to search and visualize parts that are clashing. Due to some company-specific design practises, such as modeling fasteners without visible threads to reduce processing load on the designers computer, the collision detection results yielded in high amounts of collisions that will not affect the final assembly process. The high amount of detected false collisions overflow the simulation results and drastically increase the workload of designers who are tasked to go through these results in order to identify the collisions that can actually cause problems.

This thesis has two main objectives: (1) to research if a machine learning model could be used to filter the collision detection reports to reduce the amount of falsely detected collisions and (2) to develop a software system that combines the aforementioned machine learning model, a web-based user interface, and a database solution into a single tool to reduce the workload of designers and to speed up the DFM analysis process.

This thesis consists in total of six chapters. The next chapter contains the literature review in which the main concepts behind this thesis work and the larger virtualization project the thesis is a part of are briefly explained. These concepts include Design for X, Design for Manufacturability, machine learning and version control. In the third chapter the implementation of the system itself is discussed in more detail. The fourth chapter presents the evaluation method and results, which are then discussed further in the fifth chapter. Finally, the last chapter presents the summary and conclusions of this thesis.

2. BACKGROUND

This chapter provides background for the larger project that this thesis is a part of and for the thesis itself. It explains how design for manufacturability can benefit a modern new product introduction process. In addition, some of the key concepts of machine learning and version control are discussed.

2.1. Design for X

Design for Excellence or Design for X is an engineering practice which manages guidelines on how different properties of products can be reached by considering them early in the product development process. The “X” in Design for X stands for a variable that can have one of many different values, each referring to one of the applications of DFX. These applications may feature different specifications and issues such as cost, reliability, or manufacturing, creating new terms such as Design for Cost (DFC) or Design for Manufacturability.[1]

DFX is a term first coined in the late 1970s in the context of defining design of assembly in order to streamline production costs during the manufacturing process [10]. Later, starting from the 1990s, a multitude of research papers on DFX and its applications have been published, especially regarding manufacturing which was the main theme of DFX for a long time [2]. Since then, DFX has shifted from revolving around manufacturing to considering the design process as a whole and accounting for new product design and life phase challenges arising from the development of the surrounding world [1]. These new design and life phase challenges include problems such as environmental concerns or product recyclability, which have resulted in terms such as Design for Environment [1].

2.1.1. DFX Applications

The "X" in DFX can have two meanings: it can either mean excellence in the sense that the design process aims to create a product that is overall excellent and that the design is complete, or the more common interpretation where X refers to all the desirable features or objectives that the product should have[11]. The different objectives that the X represents can be categorized either as virtues that the product should be optimized for, such as quality or cost, or it can represent a particular phase of the product’s lifecycle such as manufacturing or disposal [1]. Examples of both categories are presented in Table 1.

Table 1. DFX virtues and life phases. Adapted from [1]

Design for Environment
Design for Quality
Design for Maintainability
Design for Reliability
Design for Cost
Design for Manufacture and Assembly
Design for End-of-life
Design for Disassembly
Design for Recycling
Design for Supply chain

DFX virtues are not something that the product itself should possess, but rather measures for checking how well the product fulfills the given virtue. These virtues are more of an extension for the products primary purpose and are there to satisfy customers' or stakeholders' needs in such areas as cost, appearance or recyclability. [1]

DFX life phases on the other hand are something that help make sure that the entire product life cycle is taken into consideration when setting requirements for the product. DFX life phases are there to help designers make sure that they can understand that the implications of their design choices affect the whole product life cycle as choices made in earlier life phases affect subsequent life phases. [1]

Each DFX virtue or life phase has its own specific purpose, such as Design for Manufacturability's purpose to reduce production cost [1]. Different DFX applications may partly or fully have the same purpose with other applications and they can also be used together in aiding of achieving said purpose [1]. DFX applications may also overlap with each other and in fact block other applications from achieving their goals [1]. For example, trying to reduce environmental impact by implementing Design for Environment approaches can negatively affect the Design for Cost's goal to reduce cost if, for example, more expensive, environmental-friendly materials are used [1]. Examples of applications and their purposes are presented in Table 2.

Table 2. Examples of DFX applications and their main purpose. Adapted from [1]

Design for	**Purpose**
Manufacture and Assembly	Reduced production costs
Environment	Reduced environmental impact
End-of-life	Reduced environmental impact
Disassembly	Reduced disassembly cost/time
Recycling	Better recyclability
Quality	Greater user satisfaction
Maintainability	Reduced maintenance cost
Reliability	Reduced failure rate
Cost	Reduced cost

2.1.2. Benefits of DFX

Benefits of DFX can vary based on how the DFX is implemented, but they can be categorized into three different categories, the first of which is competitiveness related benefits such as improved quality, reduced life cycle cost or time-to-market. The second category includes organizational benefits such as project management. The third category is about cost drivers of a project's life cycle, such as part count. [12]

Implementation of DFX tools will often require higher investments and more effort early in the product's life cycle, but by considering different design aspects early in the design stage companies can produce better products down the line [10].

By successfully implementing DFX the product is designed correctly and can enter the marketplace faster with fewer delays or problems, which will result in a product that meets customers' needs better, is easier to transit into manufacturing and has a lower total life cycle cost. [10]

2.2. Design for Manufacturability

Design for Manufacturability or sometimes Design for Manufacturing is a term derived from DFX where the X has been assigned the value "M" to represent process issues of manufacturing.

DFM is an engineering practice with a base idea to design products in such a way that they are easily manufacturable. Wider definitions for DFM emphasize creating products that not only are easy to manufacture or assemble, but also are marketplace winners by improving product competitiveness by means of streamlining the manufacturing process and cost. [3]

The concept behind DFM was founded with Design for Assembly (DFA), another DFX application that focuses on ease of assembly, from which DFM expands and adds on top to create guidelines for the manufacturing process as a whole [10]. While DFM and DFA are separate applications, they do share some commonalities and can sometimes be seen combined together as Design for Manufacturing and assembly, or DFMA for short [13]. Both DFM and DFA were derived from the same recognition

that the production cost is directly linked to the product's design. Whereas DFA is also concerned with the product's structure, DFM is only concerned about the component level design [1].

2.2.1. DFM Guidelines

At its simplest, DFM support is provided in the form of general design guidelines that state which features or objectives should be avoided or included in a product's design. In addition to general guidelines, different manufacturing processes have their own separate guidelines concentrating on the problems of that specific process. All of these guidelines aim for the same goal of creating products that are easy to manufacture with a lower cost. The general manufacturing guidelines are presented in the list below:

- assemblies should be designed with minimal number of parts
- use of standard parts when possible
- use of modular design
- multifunctional parts
- use same standard parts for multiple products
- maximum surface roughness and tolerance
- avoid secondary processes
- use materials that are easy to manufacture
- minimize the handling of parts. [2]

As can be seen from the list, the general DFM rules focus on streamlining the manufacturing process as a whole by reducing secondary and parallel manufacturing processes, such as welding or surface finishing, and by minimizing part types and quantities while maximising part usage in different designs and functions. By these means the manufacturing process and the product itself can be kept standardized and uniform, which will lead to reduced manufacturing cost. [2]

2.2.2. DFM and Simulation

In addition to the general and process specific-guidelines, different DFM tools can be implemented to ensure successful DFM process. These tools are divided into three different categories:

- multi-functional teams
- use of computer-aided design
- use of a variety of analytical techniques and methods. [3]

While usage of multi-functional teams and computer-aided design has been standard practice in product design for a number of years, DFM analysis did remain a manual task for a long time. As a manual task DFM analysis is time consuming and a difficult process and prone to human errors, which evoked the need for computer-aided analysis and simulation software. [4]

As DFM has evolved since 1970s, so have the CAD systems, and a variety of feature-based CAD software are now available and have become standard practice in the DFM field. These feature-based software aid in many DFM-related tasks such as tolerance specification, collision detection and assembly design, and overall shorten the time taken in DFM analysis. [4]

Feature-based software systems range from simple solutions that display information of relevant guidelines for the designer during the design process, to more elaborate and comprehensive systems that can analyze and simulate the whole design, highlight possible errors that contradict with the guidelines and offer direct solutions for fixing the found design flaws. [4]

Benefits of feature-based systems extend further than just reducing the workload of the analysis process. Feature-based software allows the analysis to be done by the designers themselves [14]. This removes the need for separate DFX specialists and allows designers to get instant feedback from their work and then implement the required changes in their design, thus speeding up the redesign process [4]. Feature-based systems can also make sure that every designer is working with the same guidelines to produce consistent designs [14].

2.2.3. Benefits of DFM

The main goal of DFM is to drive down the cost of production, but other benefits can be achieved simultaneously when DFM is implemented successfully. Different benefits of DFM are listed below:

- improved design quality
- increased product performance
- higher yield
- reduced manufacturing costs
- reduced time-to-market
- reduced manufacturing time
- reduced product development cycle. [2]

2.3. Machine Learning

There is a common misconception that the terms artificial intelligence and machine learning (ML) have the same meaning and are often used interchangeably, when in

reality artificial intelligence is a far broader concept and machine learning is just a subfield of AI referring to computing algorithms that are used to learn from input data [5]. Machine learning can be used to solve problems such as text or document classification, computer vision applications and fraud detection [15].

Machine learning can be further divided into three different subcategories, supervised, unsupervised and reinforcement learning [6] as shown in Figure 2. From this thesis paper's perspective, the first two categories, supervised and unsupervised learning, are the most important and are described in more detail.

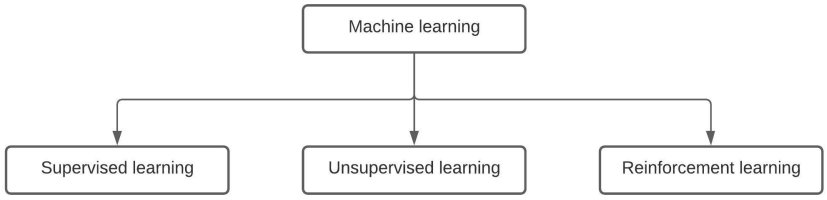

Figure 2. Machine learning subcategories.

2.3.1. Supervised Learning

In supervised learning the algorithm is supervised by giving the input data in labeled form containing information of which category or class the data point belongs to. This allows the algorithm to learn the desired outputs for each data point and draw conclusions on how different inputs affect the desired outcome.The two main types of supervised learning problems are classification and regression [16].

Classification aims to predict a categorical class label for an unlabeled data point based on past observations of the input data. Classification can be further split into binary or multiclass classification depending on the number of labels being predicted. Binary classification refers to a problem in which predictions are made between exactly two different classes whereas multiclass classification is problem with three or more classes. [6]

Regression on the other hand is a problem where real, continuous values, such as prices, are predicted based on the input data [15]. Regression aims to find a relationship between input variables and a continuous response variable that allows the outcome to be predicted [17].

2.3.2. Unsupervised Learning

Whereas supervised machine learning algorithms receive labeled data to learn patterns, unsupervised algorithms do not. Unsupervised algorithms learn different patterns by observing the input data and conclusions are made solely based on the recognized patterns [18]. Unsupervised learning includes algorithms such as clustering and dimensionality reduction [19].

Clustering refers to a problem of partitioning a set of data points into different homogeneous subsets or groups and separating them without any prior knowledge of which group the data points belong to [15, 17].

Dimensional reduction, as its name implies, aims to transform an initial representation of items into a lower-dimensional form, i.e. to reduce dimensions. Dimensional reduction compresses the data into a lower dimension while keeping most of the relevant information and is often used for data visualization and noise removal. [17]

2.3.3. One-Class Classification

One-class classification is a machine learning method that tries to identify data points of a specific class among all data points when trained with a data set containing data only of that one class, as opposed to the traditional classification problem where the data points are separated into two or more classes. Typically in a one-class classification scenario the data set is imbalanced, meaning that the data set contains high amount of data points in one class, and just a few in another. [6]

One-class classification algorithms are used to solve problems such as outlier detection and novelty detection [6].The main difference between outlier and novelty detection is that outlier detection tries to detect outliers inside the given training data set by finding similar patterns in that data set [20]. Novelty detection is very similar to the outlier detection, with the main difference being that novelty detection tries to find outliers in a new data set based on the training data [20].The most widely used one-class classification algorithms are one-class Gaussian, one-class kmeans, one-class kNN and one-class support vector machine (SVM) [6]. In this thesis work the one-class support vector machine is used.

2.4. Version Control Systems

As the name implies, version control systems are used for managing multiple versions of a project or a file [21]. Most often the term version control is associated with storing of source code files during software development utilizing a Git-based version control system [22].

Version control systems solve the problem of storing and sharing files when more than one person is sharing the same files as they can be accessed and edited by multiple persons at the same time [23]. Version control systems also allow users to track the changes made to the files and maintain the possibility to return back to a previous specific version if needed [21].

Version control systems can be categorized as centralized (CVCS) or distributed (DVCS) version control systems, the main difference being how the files are being stored. Centralized version control systems store the files on a central server, enabling team collaboration, whereas DVCS clients fully mirror the whole repository which can be then copied back to the server in case of failure. CVCS does not have this option, and if the server fails, the whole project can be lost. [22]

For the purposes of this thesis, a centralized version control system was implemented. The main reason was that centralized system is far simpler to implement as files are stored only in one central location, in this case in the database from which the data is then retrieved to be displayed in the web-based user interface.

3. IMPLEMENTATION

In this chapter of the thesis the implemented software system is discussed in more detail. First, the main software requirements, architecture, components and the software stack of used technologies are introduced, the second part focuses on data collection and preprocessing, after which the implementation of each software component is explained in more detail.

3.1. Software Requirements

Before starting software development, user stories were collected to gather more information and to map out what features should be implemented for this software system to fix shortcomings of the simulation method. User stories were collected from professional 3D designers, DFX specialists as well as project leads that are involved with different projects regarding virtualization of the design and NPI process.

The most important feature to be implemented was the ability to reduce the number of false collisions in the simulation results. In addition, a user interface for easy access to the filtering system and previously filtered reports should be implemented. A solution for storing the reports and tracking them based on products, product versions and simulation date was needed. A list of features derived from the collected user stories is presented in Table 3 and implementation of these features is further discussed in the following section.

Table 3. Software requirements

No.	**Requirement**
R1	Ability to reduce the amount of false collisions in simulation results
R2	Ability to browse products, versions and reports attached to them
R3	Ability to upload new simulation results
R4	Ability to add new products and product structures
R5	Ability to track simulation results based on product and product version
R6	Ability to store product structures and versions
R7	Reports should be made more readable

3.2. Software Architecture

Based on the list of required features derived from the collected user stories, a more detailed software architecture was defined. To accommodate all the needed features the software system would consist of three main components: the frontend that is the graphical user interface visible for the end users, the backend where the machine learning and other logical components reside, and the database where the collision report and machine learning data is stored. A high-level software architecture diagram is presented in Figure 3 which shows the main software components and data flow between them.

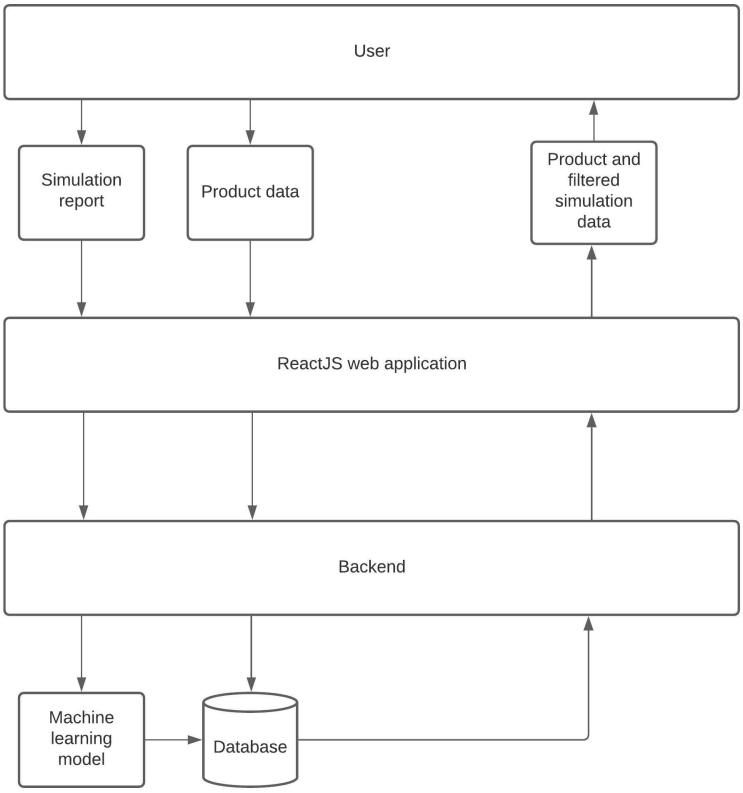

Figure 3. High-level software architecture diagram.

As the different software components include avariety of different features ranging from web-based user interface to machine learning and database solutions, the software stack needs to be selected accordingly, so that it can accommodate all these features. For these reasons the so-called FReMP stack (Flask, ReactJS, MongoDB and Python) was chosen as it allows easy integration of needed software subsystems into one full-stack application [24].

FReMP stack offers the benefit of using the Python programming language to build the backend along with the Flask web framework to service a web-based user interface, allowing easy integration with other Python libraries, most importantly Scikit-learn and Pandas, to handle tasks such as machine learning operations and data handling, while still offering the ability to build modern user interface solutions using the ReactJS JavaScript framework. [25, 24]

3.2.1. Frontend

The frontend, or the user interface, is the only component that is visible and directly accessible for the users. All user interactions such as uploading new simulation reports to filter or browsing existing reports are all done through the user interface. User actions are carried out from different views of the user interface and each action is communicated to the backend using REST API calls to trigger different functions in

the backend. The frontend was implemented with ReactJS which is an open source JavaScript library maintained by Facebook and specially developed for building user interface solutions [26]. Implementation of the frontend is further discussed in section 4.5.

3.2.2. Backend

In this implementation the backend has three main tasks:

- serve frontend components through REST API endpoints
- handle machine learning operations and other logic
- communicate with database.

Serving the frontend components was done with Flask which is a web framework written in Python. More precisely Flask is a micro web framework as it does not include any extra features such as database abstraction layer or form validation that can be found in some other, more complex web frameworks. This allows users to choose the features they want to implement on their web service by installing extensions supported by Flask to handle these tasks in a way as if it was an inbuilt Flask function. [27]

For this thesis work's purposes Flask was chosen. It being a micro framework the setup is quick and the development process can be kept fast and lean when adding more features. As Flask is written in Python, it offers the benefit of easy integration with other backend components, also written in Python, for handling machine learning operations and data manipulation, making it simple to access them through the web-based user interface via different REST API endpoints. The Different API endpoints and descriptions of their functions are displayed in Table 4.

Table 4. Explanation of REST API endpoints

Method	**Route**	**Description**
Get	/app	Return static home page
Get	/app/products	Get list of product names
Get	/app/reports/ <productname>	Get list of versions attached to product name
Get	/app/reports/ <productname>/ <versionname>	Get list of reports attached to structure
Get	/app/reports/ <productname>/ <versionname>/ <reportname>	Get the report data attached to report
Get	/app/reports/<all	Get list of all reports in the database
Get	/app/reports/all/ <reportname>	Get the report data
Post	/app/upload	Post a new report to system and filter it with ML operations
Post	/app/uploastucture	Upload product structure into the system

For data handling and manipulation the Pandas library was used. Pandas is a open-source software library developed for data analysis and manipulation. Pandas aims to provide fast and convenient ways to work with structured data and supports multiple data formats such as tabular data with heterogeneous columns, similar to an Excel spreadsheet. Pandas offers multiple tools for manipulating the data such as splitting, merging, and reshaping data sets, replacing missing values and more.[28]

For this thesis work, Pandas offersa convenient way to handle the data, as it is originally in the from of Excel spreadsheets and thus can be directly imported as Pandas data frame using inbuilt functions. Pandas also has great compatibility with sklearn as Pandas data frames can be directly utilized for machine learning operations. Pandas' data frames are also compatible with the MongdoDB database solution, which allows them to be easily saved and retrieved to and from the database.

Machine learning operations were implemented by using the Scikit-Learn library. Scikit-Learn, or sklearn for short, is a free to use software library used to implement machine learning solutions with the Python programming language. Sklearn includes a multitude of machine learning tools for both supervised and unsupervised learning such as classification, clustering and anomaly detection. These tools include machine learning methods such as support vector machines, decision trees and random forest classifiers. Sklearn also includes functionality for data preprosessing and model performance evaluation. [29]

In this thesis a classification system was implemented by leveraging sklearns one-class support vector machine for novelty detection to label collisions based on previously collected collision data. For preprocessing, sklearn was used for splitting

the data set into training and testing sets and for encoding categorical values by using sklearns one-hot-encoder.

3.2.3. Database

A database was needed for storing machine learning data and relevant information about products, their structures and reports attached to them to create a version control system. The database selected in this thesis work is MongdoDB which is a free to use, source available, document-based database solution. MongoDB is a NoSQL database system, meaning it can store data without a predefined schema, unlike relational database solutions, which allows a fluid and convenient way to store data in a JSON-like format. [30]

The database is separated from the backend and is run as it own network instance and is accessed only from the backend using PyMongo library functions. PyMongo [31] is a Python library which contains tools for working with MongoDB and allows easy read and write operations to be executed with Python scripts. PyMongo is also compatible with Pandas' data frames which can directly be written to the database without formatting or predefined schema.

3.3. Data Collection and Preprocessing

3.3.1. Data Collection

For the machine learning operations, data needed to be generated for training the algorithm to differentiate between false and real collisions. For data generation a simulation software was used to analyze CAD files and to export simulation results of multiple subassemblies across multiple products to produce diverse training data to ensure good recognition capability down the line.

Exported simulation results were contained in multiple Excel spreadsheet files. First these files containing collision detection results had to be gone through manually and each collision labeled to be true or false. After manual checking, these files were combined into one Pandas data frame for easy data manipulation and then empty, unnecessary columns were removed, resulting in a data frame containing only features that can provide meaningful data for the machine learning operation. The different features and their descriptions are presented in Table 5.

Table 5. Description of features in the original data set

Feature	**Description**	**Type**
Instance	Instance no.	Integer
Source	Source part name	String
Target	Target part name	String
Interference volume	Volume of collision	Integer

3.3.2. Data Preprocessing

After the collected data was combined into one single data frame, the data frame was cleaned and rows containing empty cells or missing values were removed. The resulting cleaned data frame was then written into the database to be accessed when training the machine learning model. The number of data points and the distribution of labels in the final data set are presented in Table 6.

Table 6. Data set size and number of labels after clean up

Description of final data set	
Total number of data points	2640
Number of false labels	2591
Number of true labels	49

After the initial cleaning of the data set there was some data manipulation needed before fitting the machine learning model with the training data. When exported from the simulation software, source and target part names are presented in a path format all the way from the top of the of the 3D model's structure hierarchy to the name of the colliding part as shown below.

$$mainassembly/subassembly/partname,$$

To successfully fit the machine learning model to the training data later, and to make the reports more readable, these names needed to be cleaned. The full path was removed from both source and target part names so that only the part name remains. This ensures that the part names can be successfully encoded later.

3.3.3. Feature Plots

To see if the different features presented in the collected data set do in fact provide any predictive value for the machine learning model, feature by feature plots were plotted. In the data set there were 5 features present: instance, source name, target name and interference volume and label. Instance number simply implies the order of the detected collision in simulation software while label is used for categorizing the collision as true or false. Neither of these holds any relevant information for the machine learning model and therefore will be excluded. Source name, target name and interference volume on the other hand are the features of interest. To see if these features hold relevant information for the machine learning model, feature plots for each combination were plotted, which are presented in Figure 4.

From these plots a good separation can be clearly seen between the purple and yellow points. This would indicate that each feature will in fact provide predictive value for the machine learning model and thus any of them don't need to be removed from the data set before fitting the machine learning model.

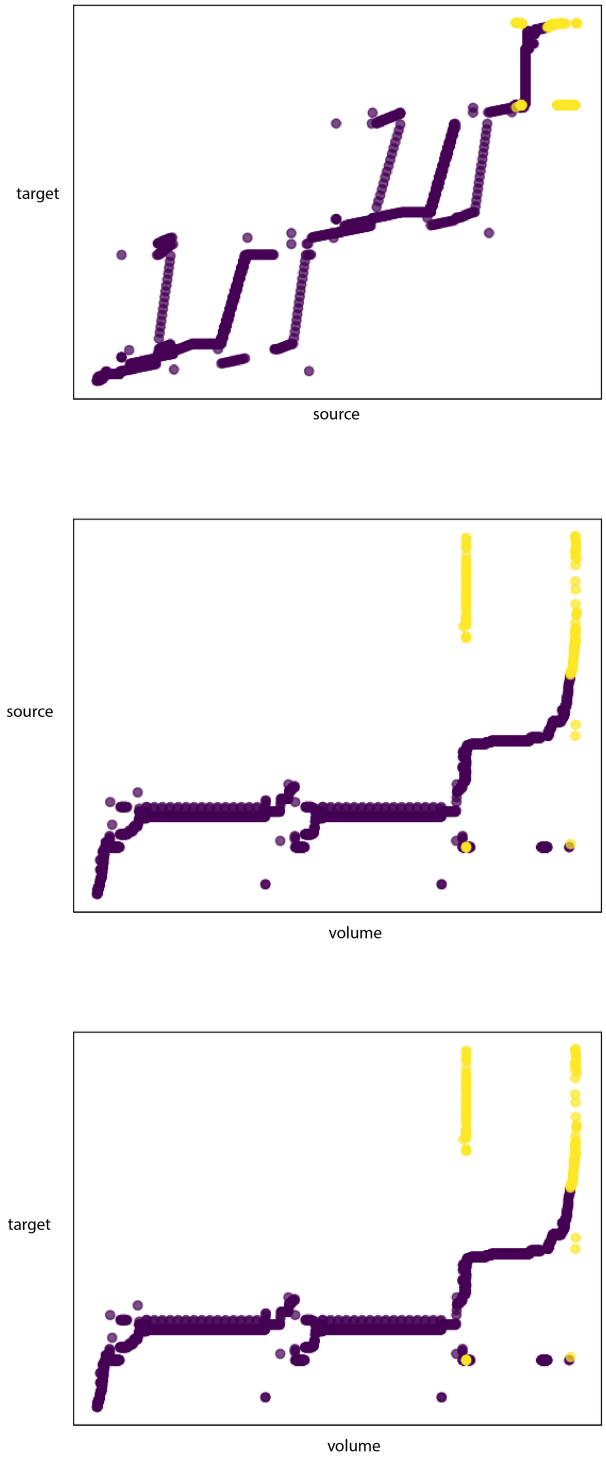

Figure 4. Feature plots.

3.3.4. Encoding

As most machine learning models can only handle numerical inputs, all fields containing categorical values needed to be either turned into numerical values or

encoded. Interference volume was originally presented in text form so all the values in that column were simply formatted into integers and units were removed. Source and target columns on the other hand needed to be encoded. For these columns one-hot-encoding was applied. By applying one-hot-encoding a binary variable is added for each unique categorical value. When applying one-hot-encoding the dimensions of the data set increased drastically as can be seen from the encoding example presented in Table 7 and Table 8.

Table 7. Example data frame before encoding

Source	**Target**
Source 1	Target 1
Source 2	Target 2
Source 3	Target 3

Table 8. Example data frame after encoding

Source1	**Source2**	**Source3**	**Target1**	**Target2**	**Target3**
1	0	0	1	0	0
0	1	0	0	1	0
0	0	1	0	0	1

3.4. Data Filtering

At first, the machine learning problem addressed in this thesis work seems rather simple: Find a way to classify detected collisions into two different categories. These two categories are: (1) Real collisions that can cause problems if they reach the physical prototype and (2) False collisions that are present in simulation data due to design practices. The problem here is that in the collected data set, there was only about 2% of real collisions presented, making the data set highly imbalanced. This critically affects the use of supervised classification algorithms such as support vector machines which rely on having a more balanced data set [32]. If a model is trained with an imbalanced data set, the classifiers often ignore the smaller class while they concentrate on classifying the larger class correctly [33]. This would create a problem with this implementation as the smaller class is the more important one.

For this reason, an unsupervised learning algorithm was implemented and trained with the normal data, i.e. the false collisions, only. For this thesis work's purposes a widely used novelty detection algorithm, a one-class support vector machine, was chosen as the machine learning model. When trained with the normal data only, the model learns the boundaries of the false collision data points and it can identify points that lie outside this boundary, classifying these as the outliers.

3.4.1. Implementation of the One-Class Support Vector Machine

For implementing the machine learning model and for testing and finding suitable parameters, the data set was split to train and test data sets utilizing sklearns train-test-split-function. The data set was separated with 80-20 split, 80% for training the model and 20% for testing the model's performance. As the model is trained with normal data i.e. false collisions only, all the true collisions were added to the test data set to evaluate the prediction capabilities of the model. The train and test data sets are described in Table 9.

Table 9. Summary of data set after train-test-split

Description of train-test-split			
Data set	**False labels**	**True Labels**	**Total**
Train	2071	0	2071
Test	518	49	567
Total	2589	49	2638

Before fitting the model, three parameters: kernel, nu and gamma needed to be defined. The kernel parameter defines the kernel type to be used in the algorithm. In this implementation the default kernel in sklearns one-class support vector machine was used. The parameter values are presented in Appendix 1. By default the kernel type is set as a non-linear kernel.

The second parameter needed is the nu-parameter which is defined as the upper bound of the fraction of training errors and a lower bound of the fraction of support vectors. To set the nu-parameter of the one-class support vector machine, the nu value was derived using the Equation (1) to be the proportion of true labels in the data set:

$$\frac{TrueLabels}{FalseLabels} = nu, \tag{1}$$

In this case, the fraction of the true labels is 0.0185 which was rounded up to 0.02, or 2%.

The magnitude of the gamma parameter was set by testing the model with different gamma parameters decrementally, starting from 0.1 with 10-fold decrements until the accuracy of the model's predictions stopped improving. The effects of different gamma values on the accuracy are presented in Table 10. As can been seen from Table 10, the variation in accuracy with different gamma values is slight, but the highest accuracy was achieved with gamma value of 0.001 which was then chosen as the parameter.

Table 10. Effect of gamma parameter to the accuracy of the model

Gamma	Accuracy
0.1	0.61
0.01	0.79
0.001	0.86
0.0001	0.81
0.00001	0.75
0.000001	0.64

After all the parameters were set, the model was fitted with the training data, and its performance tested by predicting labels for the test data set. From this test a confusion matrix was plotted, which is presented in Figure 5, and accuracy, precision and recall values calculated, presented in Table 11

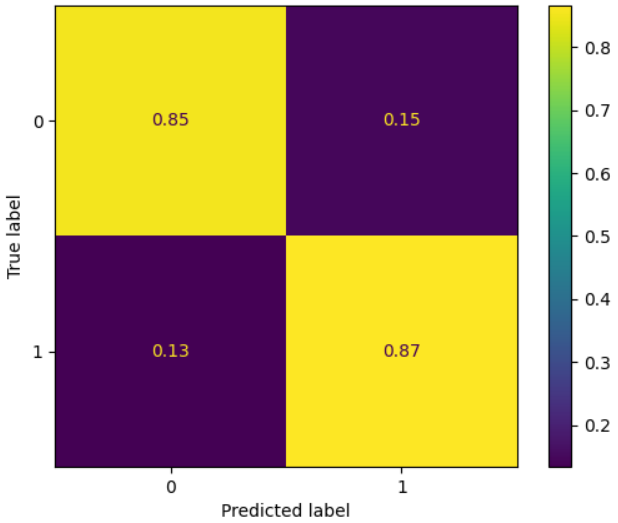

Figure 5. Confusion matrix for the model.

Table 11. Accuracy, precision and recall values for the model

Accuracy	Precision		Recall	
total	Class 0	Class 1	Class 0	Class 1
86%	86%	85 %	85%	87%

As can be seen from the confusion matrix in Figure 5 and from Table 11, the results show high percentages in each field. High overall accuracy and high precision and recall values in both classes would indicate that the model is accurate, and both classes are recognized well by the model.

3.5. Version Control System and Database

In addition to the ability to reduce false collisions in the simulation results, the requirements included ability to store the simulation reports in such a way that they are trackable by product and their versions and so that one may return to check previous simulation results in the event of an actual error finding its way to the actual physical prototype. For these reasons, a version control system and a database were implemented so that files can be stored in one central location, and are easily accessed from the web-based user interface.

The database was structured so that it can store the following information separated in four entities:

- product names
- product versions attached to each product name
- product structure attached to each product version
- simulation report attached to each part or assembly in product structure.

Each of these entities is stored in a separate database location which allows a simple database structure. To combine different entities in the database into one fluid version control system, the database is browsed based on user actions. Relevant data for each action is retrieved from the database when needed. Relations of the different entities are presented in Figure 6.

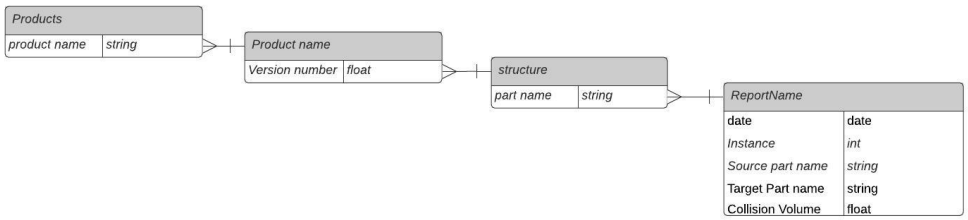

Figure 6. Database entities and their relations.

3.6. User Interface

The user interface was implemented using ReactJS. For this thesis work, a multi-page-layout was created so that uploading, and especially reading data and database navigation would be as logical and fluid as possible.

The main page of the user interface is displayed in Figure 7. On the main page the user can find the last 20 reports uploaded to the system to quickly access these reports without navigating the page further. On top of the page the user is presented with options to open a side bar for navigating products, for adding new products to the database as well as uploading new reports to be filtered.

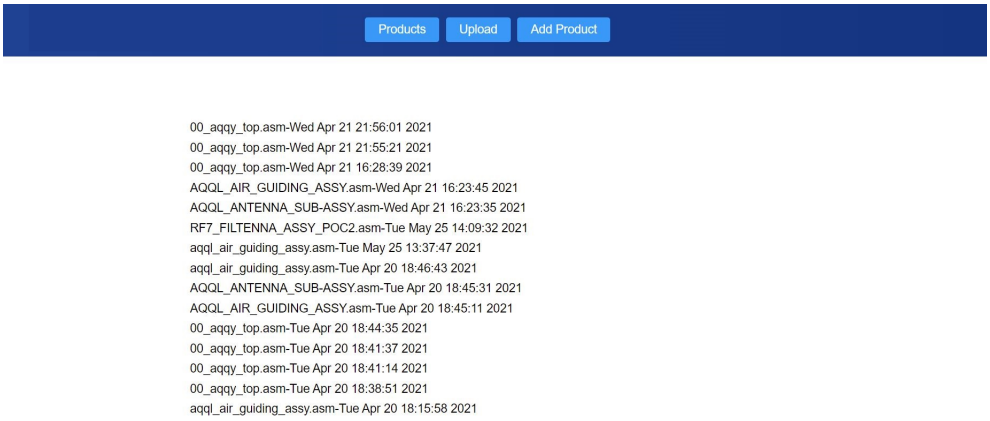

Figure 7. Main page of the user interface.

3.6.1. Inputs

For uploading data to the system two methods were implemented, both of which can be accessed from the top part of any page in the system via the Upload and Add Product buttons. Both buttons open a pop-up on top of the page. Figure 8 shows the pop-up window for uploading a new product. When adding a new product into the system the user is asked to input a product name, a version number and a product structure in the form of Excel spreadsheet which will be used to set up the new product in the database.

As can be seen from Figure 9, when uploading new reports to the system no additional information is needed, only the exported excel spreadsheet of the simulation results. The uploaded report is automatically filtered and written into the database and is instantly accessible via the user interface. The report is automatically stored in the right place and displayed when browsing the reports of the product structure in which the report is included.

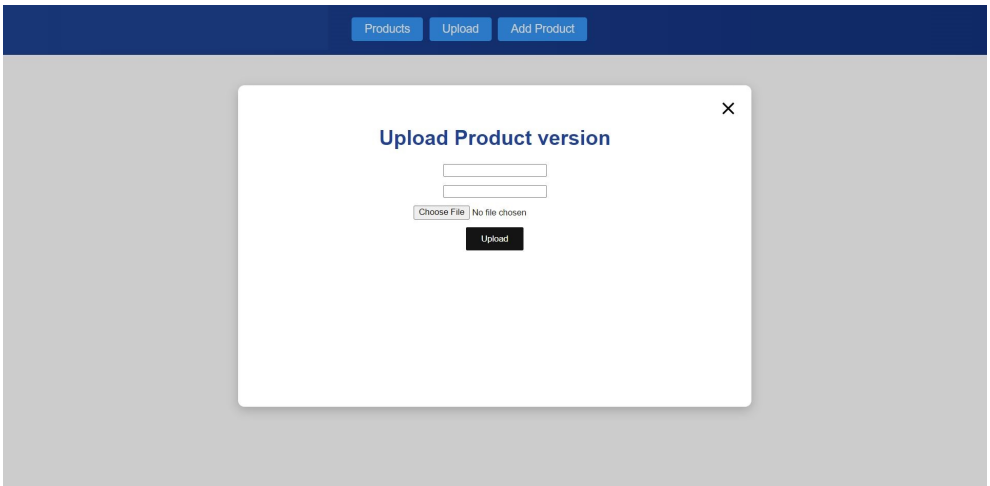

Figure 8. Product upload view.

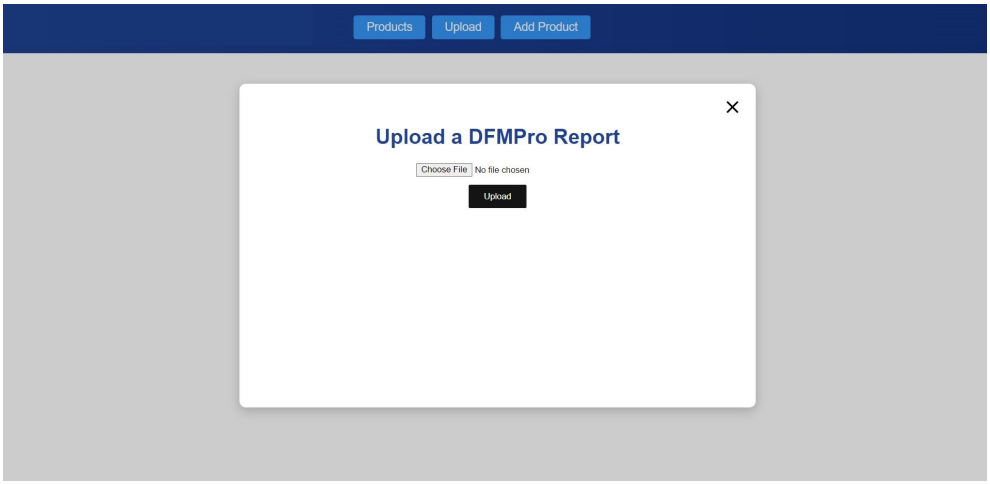

Figure 9. Report upload view.

3.6.2. Accessing Data

The page layout was designed to be as easy and as intuitive to navigate as possible. For that reason the page layout is structured much like the database is, as can be seen in Figure 6 in the previous section. Similar database structure and page layout allow easy data retrieval as only the relevant data for each navigational step needs to be retrieved from the database each time, which can be done with simple PyMongo operations. Figure 10 displays the navigation process from the front page all the way to a single report.

Figure 10. Page navigation.

From front page the list of products (as shown in Figure 11) can be opened to the sidebar by clicking the Products button in the top bar of the page. By clicking any of the product names on the list, the different versions attached to that product are displayed as shown in Figure 12. When the version is selected the user is presented with all the reports in the system that correspond to that product version based on the structure uploaded earlier, as shown in Figure 13. Finally, when a single report is selected the contents of the filtered report are displayed as shown in Figure 14.

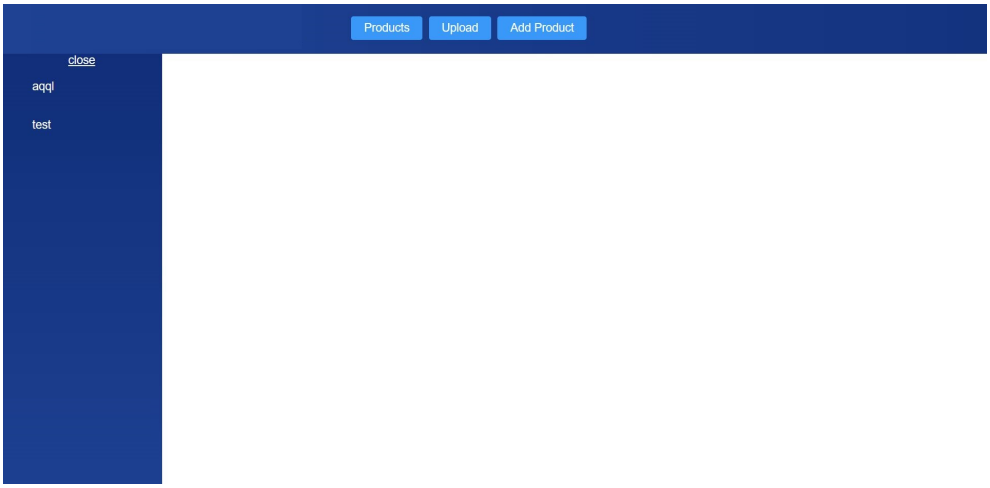

Figure 11. Product list view.

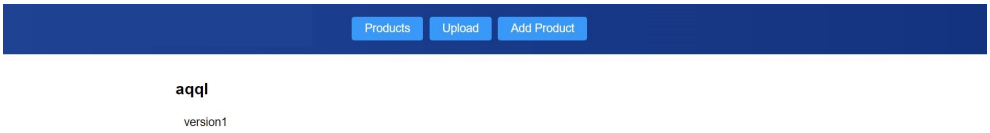

Figure 12. Version list view.

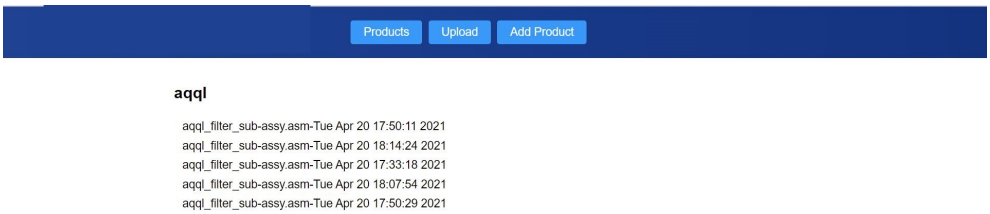

Figure 13. Report list view.

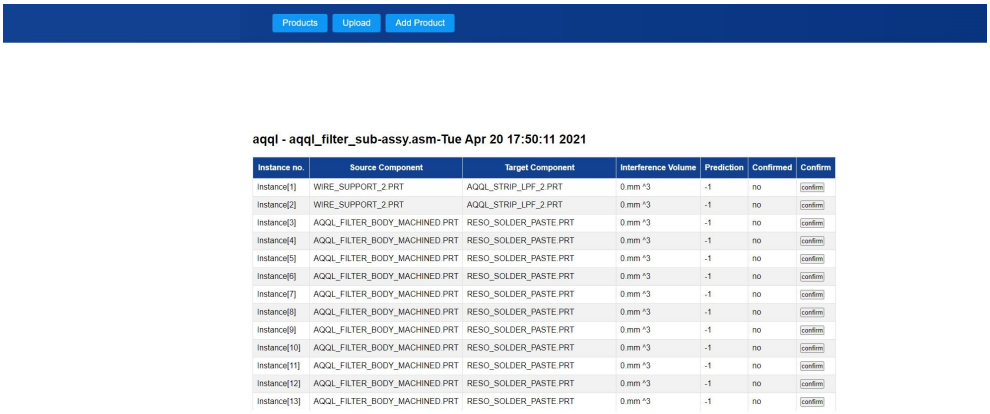

Figure 14. Single report file view.

4. EVALUATION

In this chapter evaluation methods and results are presented. The evaluation of the system is based primarily on the performance of the implemented machine learning model as it is the main function of this implementation and its performance can be quantified and visualized. Other benefits of the software implementation are presented in the discussion chapter.

4.1. Evaluation Methods

To further evaluate the machine learning model's ability to recognize and label collisions, the model's performance was tested against four different subassemblies from four different products that were modified by professional a 3D designer to include collisions that emulate real-life design flaws. Some of the purposely made collisions were severe design errors, such as parts protruding into each other, and some more slight such as a slightly misplaced fastener.

The four subassemblies were chosen in a way that three out of four subassemblies tested were from products that had their subassembly simulations in the training data and one product was totally new and no data from that product was present in the training data set to evaluate how well the model can generalize to new part names and different naming practices of parts between products.

After the modified 3D models were analyzed with the simulation software, the results were exported and gone through manually and each collision labeled accordingly to be either true or false collision.

Finally, for each report the labels were predicted with the implemented one-class support vector machine and these labels then compared to the manually entered correct ones. Confusion matrices were plotted for each report to further visualize the performance. Accuracy, precision and recall values were calculated based on these matrices.

4.2. Evaluation Results

Plotted confusion matrices for all four tested subassemblies are presented in Figure 15.

In addition to the confusion matrices, accuracy, precision and recall values were calculated for each tested subassembly report which are presented in Table 12 with average values for each category as well.

As can be seen from the plotted confusion matrices in Figure 15 and from Table 12 the model reached as high as 87% accuracy, while correctly identifying as many as 75% of the false collisions without losing any of the real collisions. The subassemblies one and two had the highest success rate and these two subassemblies happen to be the ones from which there was the most previous collision data present in the training data set. For subassemblies three and four, the results were not that great, higher amounts of real collisions were lost while only around 50% of the false collisions were detected correctly.

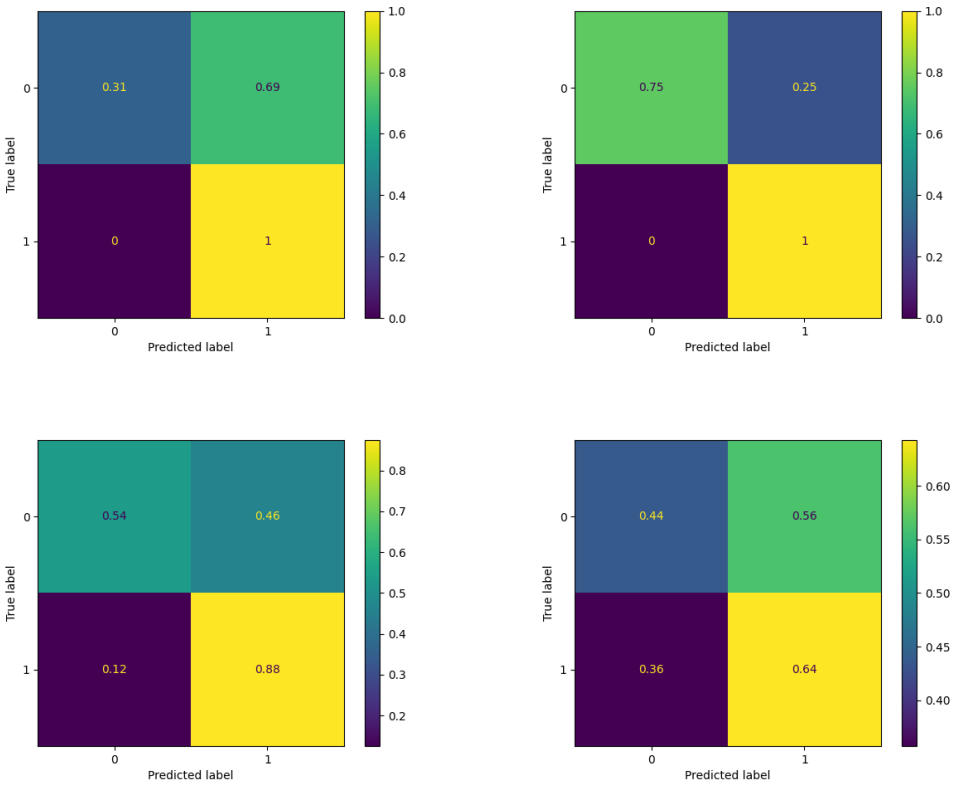

Figure 15. Confusion matrices of tested subassemblies.

Table 12. Evaluation results

Report	**Accuracy**	**Precision**		**Recall**	
		Class 0	Class 1	Class 0	Class 1
1	65%	100 %	60 %	31 %	100%
2	87%	100%	80%	75%	100%
3	71%	81 %	65%	54%	88%
4	54%	55 %	49 %	44 %	64%
average	69%	84%	64 %	51 %	88%

This would indicate that the model's performance is highly dependent from similarities between training and testing data, as can be expected. Overall these results indicate that machine learning and the implemented one-class support vector machine can be utilized for filtering simulation results and with the right amount and type of training data decent results could be expected.

5. DISCUSSION

The implemented one-class support vector machine showed great promise in detecting false collisions amongst the simulation data. When initially tested with train-test-split during the development process, as high as 85% accuracy was achieved with a relatively small training data set of only about 2650 samples. Of course splitting the training data set into two and using it as testing data does not mirror the real-life situation and should be used only for initial testing of the model.

In a more real-life like scenario the model's performance was more inconsistent, and even as high as 87% overall accuracy and as many as 75% of the false collisions detected correctly, without loss of any real collision was achieved. This was only in the most optimal scenario where the training data set contained collision data from this same product. When tested against products for which there was less collision data in the training data set, the performance was drastically degraded and only around 50% accuracy, precision and recall values were achieved. This implies that the model's predictions in this case were in practice guesswork. Of course it should be noted that the current data set used in training the model was relatively small, which could at least partly explain the poor performance in some of the tests.

Overall, these results would indicate that while the implemented one-class support vector machine could in fact be used as a tool to filter the simulation reports, much more training data would be needed to be collected from a wider range of different products to provide enough diverse training data for the model to work across the whole product range with high enough accuracy to be taken in use as an everyday designer tool.

Otherwise the system functions as intended, all the necessary components were implemented and requirements were met which are displayed in Table 13.

Table 13. Software requirements

No.	**Requirement**
R1	Ability to reduce the amount of false collisions in simulation results
R2	Ability to browse products, versions and reports attached to them
R3	Ability to upload new simulation results
R4	Ability to add new products and product structures
R5	Ability to track simulation results based on product and product version
R6	Ability to store product structures and versions
R7	Reports should be made more readable

The main requirement for this implementation was the requirement R1. It was fulfilled by implementing the one-class support vector machine for novelty detection, the performance of which was discussed in Chapter 4. To fulfill requirements R2, R3, R4, R5 and R6, suitable frontend components with corresponding backend and database methods were implemented. The implemented system allows users to upload new simulation results and to add new products and product structures into the system. The reports are stored in a database where they can be retrieved later via the implemented user interface. Requirement R7 was fulfilled by removing the path format from source and target part names which can make the reports difficult to read.

6. CONCLUSION

The goal of this thesis was to create a software system tool to support design for manufacturability in a new product introduction process. Requirements for the tool were collected from professional 3D designers and DFX project leads, and the software was implemented based on the collected requirements. The main objective of the tool was to implement a system that 3D designers could use for filtering and storing simulation results. Filtering of the reports was implemented by utilizing machine learning and a one-class support vector machine model for novelty detection.

The machine learning model implemented in this work shows promise to be actually used as a tool for filtering simulation reports. When evaluated with real-life examples, the model was able to recognize and label collisions in some of the tested assemblies correctly without losing any of the actual collisions that could cause errors in final assembly. But further development and data collection is still needed for the system to be used as an every day tool with high enough accuracy.

In addition to the implemented software system, this thesis includes a literature section in which essential technologies and methodologies, such as design for manufacturability and machine learning, are explained to offer background and explain the motivation for this thesis and the larger ongoing virtualization project that this thesis is made as a part of.

The solution presented in this thesis fulfills all set requirements. All required software components were implemented and the tool works as intended. Designers can use the system for filtering and uploading simulation reports to the database and are now able to track the simulation reports based on product names and versions through a convenient user interface that is easily accessible.

As stated before, the main issue with the simulation software was the fact that the results were overflown with false collisions, which increased the workload of designers going through the results and made use of the simulation data difficult. For moving from NPI to virtual NPI process, and to successfully implement DFM methodology, the simulation results need to be validatable so that they can be used in redesign when iterating the prototyping process. By using the software system presented in this thesis, designers could reduce the amount of false collisions in the simulation data, which would reduce the workload and speed up the validation process so that the simulation results could be used as intended.

While the implemented system shows great promise, in its current state the machine learning model implemented in this thesis is only a proof of concept and further development would be needed for it to be taken in use as everyday tool. The main focus of development would be to collect more training data for the model to increase its performance so that high enough prediction accuracy would be achieved.

7. REFERENCES

[1] Raymond H. (2009) Towards an integrated approach to "Design for X": an agenda for decision-based DFX research. Research in Engineering Design 21(2), pp. 123–136. DOI: https://doi.org/10.1007/s00163-009-0081-6.

[2] Chu W.S., Kim M.S., Jang K.H., Song J.H., Rodrigue H., Chun D.M., Cho Y.T., Ko S.H., Cho K.J., Cha S.W., Min S., Jeong S., Jeong H., Lee C.M., Chu C. & Ahn S.H. (2016) From Design for Manufacturing (DFM) to Manufacturing for Design (MFD) via Hybrid Manufacturing and Smart Factory: A Review and Perspective of Paradigm Shift. International Journal of Precision Engineering and Manufacturing-Green Technology 3(2), pp. 209–222. DOI: https://doi.org/10.1007/s40684-016-0028-0.

[3] Youssef M. (1994) Design for Manufacturability and Time-to-Market Part 1: Theoretical Foundations. International Journal of Operations Production Management 14(12), pp. 6–21. DOI: https://doi.org/10.1016/j.autcon.2017.12.021.

[4] Hoque A., Halder P., Parvez M.S. & Szecsi T. (2013) Integrated manufacturing features and Design-for-manufacture guidelines for reducing product cost under CAD/CAM environment. Computers Industrial Engineering 66(4), pp. 988–1003. DOI: https://doi.org/10.1016/j.cie.2013.08.016.

[5] Loy J. (2019) Neural Network Projects with Python. Packt Publishing, 11-13 p.

[6] Burkov A. (2019) The Hundred-Page Machine Learning Book. Andriy Burkov, 3-7 p.

[7] Kahraman G., Buyukozkan G. & Ates N. (2007) A two phase multi-attribute decision-making approach for new product introduction. Information Sciences 177(7), pp. 1567–1582. DOI: https://doi.org/10.1016/j.ins.2006.09.008.

[8] Buyukozkan G. & Feyzioglu O. (2004) A new approach based on soft computing to accelerate the selection of new product ideas. Computers in Industry 54(2), pp. 151–167. DOI: https://doi.org/10.1016/j.compind.2003.09.007.

[9] Carbonell-Foulquie P., Munuera-Aleman J.L. & Rodriguez-Escudero A. (2002) Criteria employed for go/no-go decisions when developing successful highly innovative products. Industrial Marketing Management 33(4), pp. 307–316. DOI: https://doi.org/10.1016/S0019-8501(03)00080-4.

[10] Kuo T., Huang S. & Zhang H. (2001) Design for manufacture and design for 'X': concepts, applications,and perspectives. Computers Industrial Engineering 44(3), pp. 241–260. DOI: https://doi.org/10.1016/S0360-8352(01)00045-6.

[11] Bralla J. (1996) Design For eXcellence. Technicraft publishers, 3-49 p.

[12] Huang G. (1996) Design for X Concurrent engineering imperatives. Springer Science+Business Media, 10-24 p.

[13] Zhenmin Y., Chengshuang S. & Yaowu W. (2016) Design for Manufacture and Assembly-oriented parametric design of prefabricated buildings. Automation in Construction 88(1), pp. 13–22. DOI: https://doi.org/10.1016/j.autcon.2017.12.021.

[14] Bralla J. (1986) Design for manufacturability handbook. The McGraw-Hill Companies, Inc, 70-85 p.

[15] Mohri M., Rostamizadeh A. & Talwalka A. (2018) Foundations of Machine Learning. The MIT Press, 1-7 p.

[16] Müller A.C. & Guido S. (2016) Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, 1-4, 25-127, 131-208 p.

[17] Raschka S. & Mirjalili V. (2017) Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow. Packt Publishing, 72-112 p.

[18] Shalev-Shwartz S. & Ben-David S. (2014) Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 1-8 p.

[19] Ghahramani Z. (2005) Unsupervised learning. In: Advanced Lectures on Machine Learning. Springer-Verlag, 72-112 p.

[20] Alla S. & Adari S.K. (2019) Beginning Anomaly Detection Using Python-Based Deep Learning: With Keras and PyTorch. apress, 17-20 p.

[21] Tsito M. (2020) Beginning Git and GitHub A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer. apress, 3-10 p.

[22] Hutten D. (2017) Learn version control with Git a step-by-step ultimate beginners guide. 9-11 p.

[23] Blischak J., Davenport E.R. & Wilson G. (2016) A Quick Introduction to Version Control with Git and GitHub. PLoS Comput Biol. 12(1), p. e1004668. DOI: https://doi.org/10.1371/journal.pcbi.1004668.

[24] FReMP-stack. URL: https://fremp.github.io/. Accessed 24.5.2021.

[25] Python. URL: https://www.python.org/. Accessed 24.5.2021.

[26] React. URL: https://reactjs.org/. Accessed 24.5.2021.

[27] flask. URL: https://flask.palletsprojects.com/en/2.0.x/. Accessed 24.5.2021.

[28] Pandas. URL: https://pandas.pydata.org/. Accessed 24.5.2021.

[29] sklearn. URL: https://scikit-learn.org/stable/index.html. Accessed 24.5.2021.

[30] MongoDB. URL: https://www.mongodb.com/. Accessed 24.5.2021.

[31] PyMongo. URL: https://pymongo.readthedocs.io/en/stable/. Accessed 24.5.2021.

[32] Dealing with Imbalanced Data A guide to effectively handling imbalanced datasets in Python. URL: https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18. Accessed 1.6.2021.

[33] Guide to Classification on Imbalanced Datasets. URL: https://towardsdatascience.com/guide-to-classification-on-imbalanced-datasets-d6653aa5fa23. Accessed 1.6.2021.

8. APPENDICES

Appendix 1 The list of parameters used in the one-class support vector machine

- kernel = "rbf"
- nu = 0.02
- gamma = 0.001